For publication on nepp.nasa.gov; proceedings of 2014 Single Event Effects (SEE) Symposium and the Military and Aerospace Programmable Logic Devices (MAPLD) Workshop.

Design for Review – Applying Lessons Learned to Improve the FPGA Review Process

Marco A. Figueiredo[1] and Kenneth E. Li[2]

[1]Orbital Sciences Corporation, 301-286-3327, marco.a.figueiredo@nasa.gov

[2]NASA Goddard Space Flight Center Code 561, 301-286-5659, kenneth.e.li@nasa.gov

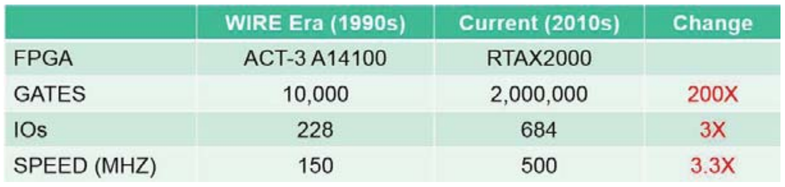
Abstract: Flight Field Programmable Gate Array (FPGA) designs are required to be independently reviewed [5]. This paper provides recommendations to Flight FPGA designers to properly prepare their designs for review in order to facilitate the review process, and reduce the impact of the review time in the overall project schedule.

Keywords: FPGA design review and process improvement.

Introduction

The investigation of the failure of NASA's Wide-Field Infrared Explorer (WIRE) mission, soon after launch on March 4, 1999, due to an FPGA design defect, indicated that the most effective opportunity to catch the design error was in the design review process [5]. Table 1 indicates the growth in complexity of the largest density FPGA utilized in the WIRE era, and a commonly utilized FPGA device today.

Table 1- Comparison of FPGA Devices



	WIRE Era (1990s)	Current (2010s)	Change
FPGA	ACT-3 A14100	RTAX2000	
GATES	10,000	2,000,000	200X
IOs	228	684	3X
SPEED (MHZ)	150	500	3.3X

The growing capacity of FPGA devices enables higher complexity designs, providing many more possibilities for design faults. Therefore, a more robust FPGA design review process is required to mitigate risks and increase reliability. Ultimately, errors caught early in the design and development phases lead to lower mission costs. Currently, high density FPGA designs are equivalent in complexity to the design of Application Specific Integrated Circuits (ASIC). Managing complexity is better accomplished with improved process and product quality assurance.

Figures 1 shows a diagram where all phases of the FPGA design development process receive equal emphasis. The goal is to have effective presence and oversight of Quality Assurance (QA) personnel; dynamic and reliable documentation; complete requirements tracking, analysis, validation, and verification; mandatory independent verification; and a robust and consistent review format across projects.

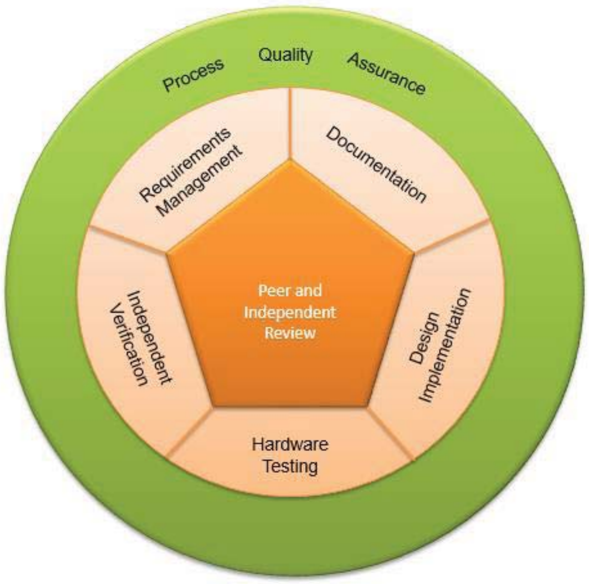


Figure 1 – FPGA Development Model

This paper presents the FPGA review process currently being utilized at NASA Goddard Space Flight Center, and provides recommendations for designers to facilitate the review process while improving the overall quality assurance of FPGA designs.

Procedures and Guidelines

The three Procedures and Guidelines listed below are used by the NASA Goddard Space Flight Center to guide the development and review of FPGA designs for space flight.

- 300-PG-8730.0.1, Digital Electronics Assurance Plan.
- 500-PG-8700.2.8-A, FPGA Development Methodology.
- 500-PG-8700.2.7-B, FPGA Design Guidelines.

The 300-PG-8730.0.1 establishes guidelines for assurance personnel to monitor FPGA design and development activities. This includes assurance personnel working in conjunction with the project to develop a Digital Electronics Assurance Plan to ensure an adequate design and that appropriate processes are followed. All flight projects should have such a plan describing the processes and methodologies adopted by the Flight Project to develop and verify the FPGA designs used in flight. The FPGA designer should understand the plan.

The 500-PG-8700.2.8-A describes good practices for the creation of a robust flight FPGA development process. It is used by the design team and project managers as a reference to create the FPGA Assurance Plan required in the 300-PG-8730.0.1.

The 500-PG-8700.2.7-B collects best design practices while using FPGA devices in flight projects. This document includes an extensive design checklist as an appendix. Designers are expected to answer the checklist items and deliver to reviewers along with a design data package for independent review. Designers should become familiar with the design review checklist prior to starting the design process.

Independent Review Process

An independent review may be performed by a single or multiple reviewers participating in a review board with the designers, verification and test engineers, software and systems engineers, and project managers.

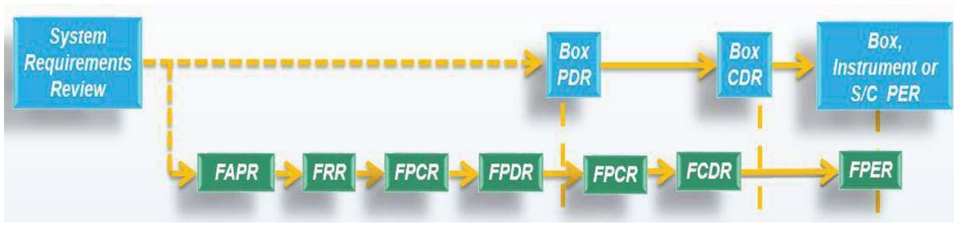


Figure 2- Independent Review Cycle

A typical independent review cycle is illustrated in the above figure. The *FPGA Assurance Plan Review (FAPR)* occurs prior to design commencement. Once the plan is approved, the *FPGA Requirements Review (FRR)* verifies that requirements are properly documented.

An *FPGA Preliminary Design Review (FPDR)* occurs prior to the box level PDR. Its main objective is to show that requirements are well understood and the design answers to the requirements accordingly.

The *FPGA Critical Design Review (FCDR)* occurs prior to the box level CDR. This detailed review happens prior to committing the design to a flight FPGA device. At this point the flight board is ready for test and waiting for the FPGA configuration. Its main objective is to assure the final FPGA product is ready to commit to flight.

The design review data package described below should be provided to the reviewers one week in advance of the FCDR. The completed **FCDR Checklist** (appendix G of the 500-PG-8700.2.7-B) is used to guide the final product assurance review. The lead reviewer manages the review meeting, collects action items, and works with designers in resolving all issues before clearing the design for flight. The lead reviewer prepares a review report and delivers to the flight project management.

The *FPGA Pre-Environmental Review (FPER)* occurs in conjunction with the box, instrument or spacecraft pre-environmental tests reviews. An FPGA independent reviewer assesses the environmental test plans and procedures and assures they are written according to the test-as-you-fly approach.

Design Review Data Package

The design review data package should contain the items described below. Preliminary reviews may be performed without a complete package. The final review before committing to a flight version of the FPGA should have all checklist items cleared by the reviewers and thus requires the complete data package.

(__) 1. FCDR review checklist,
(__) 2. Box level requirements,
(__) 3. Board level requirements,
(__) 4. FPGA requirements,
(__) 5. FPGA design specification document,
(__) 6. HDL source code including any third party intellectual property code,
(__) 7. HDL Verification Plan
(__) 8. FPGA Requirements Verification Matrix (RVM),
(__) 9. HDL simulation test benches and scripts,
(__) 10. HDL code coverage report,

(__) 11. HDL synthesis files including constraints and reports,
(__) 12. FPGA configuration/database file,
(__) 13. FPGA place and route reports,
(__) 14. FPGA Static Timing Analysis (STA) results including external inputs and outputs, internal domains, and internal domain crossings in worst case scenarios.
(__) 15. Circuit Card Assembly (CCA) schematics
    (__) 1. PDF,
    (__) 2. Native format,
(__) 16. CCA parts list,
(__) 17. CCA netlist,
(__) 18. CCA layout
    (__) 1. PDF,
    (__) 2. Native format,
(__) 19. CCA Signal Integrity Analysis (SIA) results or measurements,
(__) 20. CCA Power Integrity Analysis (PIA) results or measurements.

Review Package Organization

A project folder structure is suggested in figure 2 to facilitate the independent review. A common organizational structure helps reviewers to find the data at expected locations in the design review data package.

The README file should provide the status of the design being delivered for review. Any exceptions to the expected delivery data package or the suggested organizational structure should be noted in this file, i.e., make it clear to the reviewers where to find the files and/or files missing. This file should be used as a quick reference for designers and reviewers as to what the package represents. As the design goes through reviews, changes may be applied and a new design data package may be submitted for review. The README file keeps track of any changes between reviews.

The README_SIM file should have the name of the project and verification engineer, date, simulator tool version, a list of all the relevant files in the VERIFICATION directory with a single line description of their use, and a step by step description of how to build and run the test-bench. The independent reviewer should be able to repeat the simulation results by following directions on the README_SIM file.

```
FPGA_NAME_PRJ_NAME/
        README.TXT
        FCDR_CHECKLIST
        PRESENTATIONS/
        DATASHEETS/
        REQUIREMENTS/
        SPECIFICATIONS/
        HDL/
        VERIFICATION/
                README_SIM.TXT
                RVM
                HVP
                REPORT
                VRW
                TESTBENCH/
                CODE_COVERAGE/
        SYNTHESIS/
        PLACE_ROUTE/
        STA/
        CCA/
            PARTS_LIST
            SCHEMATICS/
            PCB/
            SIA/
            PIA/
```

Figure 2 - Review Data package Structure

FPGA Requirements

System requirements should be tracked down to the FPGA level. If level 5 represents the box level requirements in a requirements management system, then level 6 should reflect the board requirements and level 7 the FPGA requirements.

High level requirements should be flowed down to the FPGA level as early as possible and derived requirements should be documented and tracked as the design matures. All FPGA requirements should be in the requirements management system, including those described in Interface Control Documents (ICDs) and component datasheets, such as timing requirements of memory devices connected to the FPGA.

As a minimum, an **FPGA Requirements Verification Matrix (RVM)** should be created to list in a table format all the FPGA requirements with their identification number and qualify them according to: verification type (simulation(S), analysis(A), inspection(I) or test(T)); verification level (HDL(H), Card(C), Box(B), Instrument(I), Spacecraft(S)), and verification unit (simulation test-bench(TB), prototype or Engineering Design Unit (EDU), proto-flight or Engineering Test Unit (ETU), or Flight Unit (FU)). The verification type should be followed by a reference (#) to where the verification can be found, as shown in the table below. For example, a requirement being verified in a self-checking HDL test-bench should list as reference the test number reported by the test-bench showing the result of the verification of the requirement.

Table 2 - Example RVM Table Header

ID	Requirement	Verification		
		Type	Level	Unit
		S, A, I, T (#)	H, C, B, I, S	TB, EDU, ETU, FU

The requirements list should contain:

(__) 1. System level requirements applicable to the FPGA,
(__) 2. Derived requirements organized by major design functions,
(__) 3. Interface definitions extracted from ICDs.
(__) 4. Interface definitions extracted from device datasheets,
(__) 5. A table listing all FPGA pins with their names and required configuration,
(__) 6. Expected range of flight operating conditions (temperature, radiation, power)
(__) 7. Reset and clock requirements,
(__) 8. Power-up and power-down sequence requirements,
(__) 9. Memories sizes and radiation susceptibility requirements.

The Requirements Review should occur with participation of project managers, systems engineers, board designers, FPGA designers, software designers, and assurance personnel. Requirements changes and/or additions at any stage of the development process should be reflected in the requirements management system.

FPGA designs that contain either a hard or soft internal processor should also have a Software Requirements Document (SRD). They are in the domain of software assurance and are not addressed in this document.

Design Documentation

Documentation should be an on-going activity and not a post design effort. Once initial requirements are established and reviewed, an **FPGA Design Description (FDD)** document should reflect the design choices. This document should evolve along with the design. It should be continuously reviewed at peer review meetings and independent reviews for consistency with requirements and the actual design.

The design specification describes the main functions of the system in natural English language supported by block diagrams, tables, and state machine diagrams. It also describes the input and output definitions and the performance under different environmental conditions (prototype, engineering model and flight versions).

The information presented in the specification document can be referred to in the FCDR design review checklist. When the information requested in the checklist is not present in the specification document, such information will be required to be presented in the checklist. The list below is a reference to the topics to be addressed:

(__) 1. Brief system level description with a block diagram showing where the FPGA fits in the overall mission/spacecraft/instrument/box system,
(__) 2. Breakdown of the FPGA main functions with a block diagram of the internal data flow,
(__) 3. A description of each of the main functions with reference to the HDL source where it is implemented. If the function is of high complexity, provide a data flow diagram for the function,
(__) 4. A table describing the inputs and outputs as implemented with pin numbers, names and electrical configuration at startup and steady state such as speed, pull-up/pull-down, open collector, maximum/minimum voltage values, drive strength, and load capability,
(__) 5. Clock tree diagram identifying all clock domains and domain crossings,
(__) 6. Reset tree diagram identifying all sources of asynchronous and synchronous hardware and software controlled reset signals,
(__) 7. Power-up and power-down sequence descriptions,
(__) 8. Memory organization and radiation mitigation schemes including register file descriptions,
(__) 9. State machine diagrams with a description of the encoding used and the radiation effects mitigation strategy.

Designs utilizing internal microprocessors should also produce an **FPGA Software Users Guide (FSUG)** describing the hardware functions and interfaces accessible to the software engineers.

HDL Code Development

Clearly written and well commented Hardware Description Language (HDL) code facilitates independent reviews. Each HDL source file should contain a header with the names of the design, the organization and the author(s); a short functional description; and a dated list of modifications.

The latest generations of FPGA devices allow for designs that may contain thousands, if not hundreds of thousands of lines of HDL code. Designers should adhere to a collaboratively established coding style. Independent reviewers may use linting tools to check for common design practices.

It is a good practice to review the HDL code at peer code reviews prior to the independent review. A properly conducted code review yields better coding practices among designers; corrects discrepancies between requirements, code, and documentation as the design evolves; and solves interface issues among designers of blocks within an FPGA or several FPGAs and/or boards.

HDL Version Control

A version control system should be utilized at the start of the project to maintain the design and verification code bases. Besides the source code, there is a diverse set of products output by the tools utilized in the FPGA design development. FPGA place and route tools yield non-repeatable results each time they are run on the same source code base. It is thus necessary to keep all intermediary results for each major design version in case a prior implementation needs to be retrieved.

The README file in the design review data package should contain information pertaining to the version of the source code delivered and the other relevant files. It is expected that the source code delivered is the exact version used to generate the synthesis, simulation, static timing analysis, place and route, and simulation results.

Data protection is of great importance. A central location should be established for design data elements to be saved frequently and a proper back-up methodology should be implemented to guarantee recovery in the worst case situation such as fire. The design review data package should be delivered as a single compressed file through a secure data connection.

HDL Verification

The formal verification of HDL code is best approached with an independent verification engineer developing a self-checking HDL simulation test bench to verify the code against the stated requirements. An **HDL Verification Plan (HVP)** should be created to document the simulation test-bench architecture, and map requirements to verification tests. The HDL requirements should be grouped according to their essential functions and a simulation test number should be assigned to each requirement or group of requirements. The simulation test-bench should report the test number according to the RVM so that there is a direct correlation between the test and the requirement. This approach facilitates the work of the independent reviewer while asserting that the HDL test-bench validates the requirements.

The verification engineer should keep track of issues found in the HDL source code and document their resolution by the designer in a **Verification Reporting Worksheet (VRW),** or equivalent. Both positive and negative testing should be applied by verifying requirements statements (positive testing) and negating the requirements (negative testing) to verify system behavior. HDL code coverage should be collected and analyzed to provide metrics of how well the test cases in the test-bench exercise the design, including one or more methods (statement, branch, condition, variable, finite-state-machine, block, and toggle). Areas of code not covered must be justified through laboratory tests or documented reasons such as non-utilized code in third-party cores.

Simulation results should be reviewed in both peer and independent review meetings. During review, a description of the test-bench architecture and its functionality, referred to as REPORT in the review package organization should be presented. Code coverage results should also be reported with areas of code not covered properly explained.

The test environment should address the test-as-you-fly-fly-as-you-test philosophy.

Third-Party Cores

Third-party soft-core components, or Intellectual Property (IP) cores, contribute to the reduction of design time and costs. But designers should be careful as IP vendors are not likely to provide the source code for their cores. It is thus extremely important that a proper risk assessment be carried out whenever an IP core is being used in a flight FPGA design. Below are some of the questions that need to be answered:

(__) 1. Has the proper flight design guidelines been followed in the implementation of the core?
(__) 2. Has an experienced independent verifier attested the flight worthiness of the core?
(__) 3. Would the IP core vendor provide access to the source code under proper non-disclosure agreements as to allow it to be incorporated with the rest of the design, and follow the same verification and validation techniques?
(__) 4. What is the flight heritage of the core?

As some of the IP cores are provided by the FPGA manufacturers, there is a tendency to accept their flight worthiness given that the manufacturer has proven the flight worthiness of the FPGA devices. This should not be the case. The flight worthiness of each IP core should be treated independently of their origins.

Worst Case Timing Analysis

Back annotated timing simulation can be utilized to verify the synthesized HDL code, after place and route, complies with the intended design. Results are compared with the functional simulation. The worst case scenario (highest temperature, lowest operating voltage, maximum radiation, and slowest process) and the best scenario (lowest temperature, highest operating voltage, zero radiation, and best process) should be considered. Results should be reported indicating the required timing margin for the clocks, and that all signals meet timing.

While it is possible to perform a system level timing simulation including best and worst case timing information for the devices the FPGA interfaces with, a Static Timing Analysis (STA) is sufficient to verify timing closure. Timing constraints should be entered into the STA tool to facilitate internal verification of timing closure in worst and best case scenarios. A timing analysis of the external interfaces can be performed using a spreadsheet to list critical interface timing parameters and identify margins. Data should be made available to the FPGA reviewers to verify the analysis that led to the conclusion of the timing closure with required margins. A timing diagram tool can be utilized to graphically illustrate the interface timing, but it is important that the reviewers be provided with the analysis data that led to the diagrams.

Circuit Card Assembly

Designing the FPGA logic is only one part of the development process. Proper placement of the FPGA in the Circuit Card Assembly demands advanced analysis of signal and power integrity issues. These analyses can be performed prior to the CCA fabrication utilizing modeling tools, as opposed to the traditional measurement of signal and power integrity in the actual CCA.

The independent reviewer needs access to the parts list, schematics, PCB layout, and signal and power integrity analysis reports. High-density FPGAs put stress on the PCB design due to their large number of IO pins. An independent reviewer will check the CCA placement and performance of the FPGA according to established guidelines and manufacturers specifications. CCA problems may manifest only during extended hours of environmental test at the spacecraft level[4]. It is important that the review package provides the data required to perform the reviews to the detail required by the mission risk assessment.

Design Re-use

It is very common that a large percentage of a spacecraft or instrument design is re-used from prior missions. This is also the case for FPGA designs. Proven flight designs reduce mission risks. However, the environmental conditions of the current mission must be taken in consideration while assessing the usability of the design for re-use.

When a new FPGA design is re-using HDL from prior missions, then the old code is integrated into the new code base and goes through the same development and independent review processes.

When an entire flight FPGA device is being re-used, it must be assessed for its compliance with the current mission environment. An independent review requires the same data specified in the design review data package described in section 4. If a prior design review checklist is available, it facilitates the review process; otherwise the **FCDR Checklist** must be completed to assure the product is well qualified for the new mission.

Design Review Presentation

Designers should prepare an introductory presentation for the FCDR. This presentation should contain:

(__) 1. Design team organization with names, responsibilities, and contact information,
(__) 2. A block diagram showing where the FPGA fits in the overall spacecraft/instrument/box system,
(__) 3. Board level power tree diagram identifying all power sources, derived powers, and sinks,
(__) 4. Box level grounding diagram,
(__) 5. A block diagram with a breakdown of the FPGA main functions and its internal data flow,
(__) 6. Design utilization metrics,
(__) 7. Design verification strategy.

Upon completion of the introductory presentation, the FCDR Checklist is used to guide the review.

Conclusion

By development and incorporation of an efficient and effective process, there is a significant value added with a minimal cost impact. The suggestions presented in this paper were collected over years of experience reviewing flight FPGA designs. At first, they might look as excessive and costly as the number of independent reviews and documents requested increase the design and development time. This is the case only the first time this approach is adopted. Once the documentation infrastructure is built for one design and designers become familiar with the review process, the improved process actually reduces development time while increasing design quality. Process quality improvement reduces time devoted to debugging, which could become expensive if the anomalies happen during the spacecraft Integration and Test (I&T) phase.

The more developers follow the guidelines presented here, the more efficient the independent design review process becomes. As a result, the independent reviews create less of an impact in the overall development schedule while producing high quality flight implementations and reducing mission risk.

Acknowledgment

The content in this article is the result of work supporting the following flight programs: **GOES-R**: Geostationary Operational Environmental Satellite-R; **JPSS FF1**: Joint Polar Satellite System Free Flyer-1; **MAVEN**: Mars Atmosphere and Volatile Evolution; **MOMA**: Mars Organic Molecule Analyzer; **OSIRIS-REx**: Origins Spectral Interpretation Resource Identification Security - Regolith Explorer; and **TIRS**: Thermal Infrared Sensor.

The NASA Goddard Space Flight Center Flight Data Systems and Radiation Effects Branch, and the NASA Electronic Parts and Packaging (NEPP) program also provided support for this work.

References

[1] 500-PG-8700.2.8-A, NASA Procedures and Guidelines, Field Programmable Gate Array (FPGA) Development Methodology.

[2] 500-PG-8700.2.7-B, NASA Procedures and Guidelines, FPGA Design Guidelines.

[3] 300-PG-8730.0.1, Digital Electronics Assurance Plan.

[4] Figueiredo, M. “GLORY Project Lessons Learned on FPGA Assurance,” NASA Goddard Space Flight Center.

[5] Barrowman, James S. “The WIRE Case Study,” NASA Academy of Program and Project Leadership (APPL).